NIST Special Publication 800-38B

DRAFT Recommendation for Block Cipher Modes of Operation: the RMAC Authentication Mode

Methods and Techniques

Morris Dworkin

October 18, 2002

Abstract

This Recommendation defines an authentication mode of operation, called RMAC, for a symmetric key block cipher algorithm. RMAC can provide cryptographic protection of sensitive, but unclassified, computer data. In particular, RMAC can provide assurance of the authenticity and, therefore, of the integrity of the data.

KEY WORDS: Authentication; block cipher; cryptography; encryption; Federal Information Processing Standard; information security; integrity; mode of operation.

Table of Contents

Table of Figures

1 Purpose

This publication is the second part in a series of Recommendations regarding modes of operation of symmetric key block cipher algorithms.

2 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Computer Security Act of 1987 (Public Law 100-235) and the Information Technology Management Reform Act of 1996, specifically 15 U.S.C. 278 g-3(a)(5). This is not a guideline within the meaning of 15 U.S.C. 278 g-3 (a)(5).

This Recommendation is neither a standard nor a guideline, and as such, is neither mandatory nor binding on Federal agencies. Federal agencies and non-government organizations may use this Recommendation on a voluntary basis. It is not subject to copyright.

Nothing in this Recommendation should be taken to contradict standards and guidelines that have been made mandatory and binding upon Federal agencies by the Secretary of Commerce under his statutory authority. Nor should this Recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, the Director of the Office of Management and Budget, or any other Federal official.

Conformance testing for implementations of the modes of operation that are specified in this Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment of the Government of Canada. An implementation of a mode of operation must adhere to the requirements in this Recommendation in order to be validated under the CMVP. The requirements of this Recommendation are indicated by the word “shall.”

3 Introduction

This Recommendation specifies an algorithm, RMAC [1], that can provide assurance of data origin authentication and, hence, assurance of data integrity. In particular, RMAC is an algorithm for generating a message authentication code (MAC) from the data to be authenticated and from an associated value called the salt, using a block cipher and two secret keys that the parties to the authentication of the data establish beforehand. One party generates the MAC and provides the MAC and the associated salt as the authentication tag; subsequently, any party with access to the secret keys may verify whether the received MAC was generated from the received data and the received salt. Successful verification of the MAC provides assurance of the authenticity of the data, i.e., that it originated from a source with access to the secret keys. Consequently, successful verification of the MAC also provides assurance of the integrity of the data, i.e., that it was not altered after the generation of the MAC.

A MAC is sometimes called a cryptographic checksum, because it is generated from a keyed cryptographic algorithm in order to provide stronger assurance of data integrity than an ordinary checksum. The verification of an ordinary checksum or an error detecting code is designed to reveal only accidental modifications of the data, while the verification of a MAC is designed to reveal intentional, unauthorized modifications of the data, as well as accidental modifications.

Because RMAC is constructed from a block cipher algorithm, RMAC can be considered a mode of operation of the block cipher algorithm. The block cipher algorithm shall be approved, i.e., specified or adopted in a FIPS or a NIST Recommendation; for example, FIPS Pub. 197 [2] specifies the AES algorithm, and FIPS Pub. 46-3 [3] adopts the Triple DES algorithm.

FIPS Pub. 198 [4] specifies a different MAC algorithm, called HMAC, that is also appropriate for the protection of sensitive data. Because HMAC is constructed from a hash function rather than a block cipher algorithm, RMAC may be preferable for application environments in which an approved block cipher is more convenient to implement than an approved hash function.

4 Definitions, Abbreviations, and Symbols

4.1 Definitions and Abbreviations

Approved	FIPS approved or NIST recommended: an algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation.
Authenticity	The property that data indeed originated from its purported source.
Authentication Mode	A block cipher mode of operation that can provide assurance of the authenticity and, therefore, the integrity of data.
Authentication Tag (Tag)	A pair of bit strings associated to data to provide assurance of its authenticity: the salt and the message authentication code that is derived from the data and the salt.
Bit	A binary digit: 0 or 1.
Bit String	An ordered sequence of 0's and 1's.
Block	A bit string whose bit length is the block size of the block cipher algorithm.
Block Cipher	See forward cipher function.
Block Cipher Algorithm	A family of functions and their inverses that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.

Block Size	The number of bits in an input (or output) block of the block cipher.
Cryptographic Key	A parameter used in the block cipher algorithm that determines the forward cipher function.
Data Integrity	The property that data has not been altered by an unauthorized entity.
Exclusive-OR	The bitwise addition, modulo 2, of two bit strings of equal length.
FIPS	Federal Information Processing Standard.
Forward Cipher Function	One of the two functions of the block cipher algorithm that is determined by the choice of a cryptographic key.
Initialization Vector (IV)	A data block that some modes of operation require as an initial input.
Message Authentication Code (MAC)	A cryptographic checksum on data that is designed to reveal both accidental errors and intentional modifications of the data.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm.
Most Significant Bit(s)	The left-most bit(s) of a bit string.
Nonce	A value that is used only once within a specified context.
RMAC	The name of the authentication mode that is specified in this Recommendation.
Salt	A parameter of an algorithm whose role is to randomize the value of another parameter.

4.2 Symbols

4.2.1 Variables

b	The block size, in bits.
k	The key length for the block cipher.
m	The bit length of the RMAC MAC.

n	The number of data blocks in the padded message.
r	The bit length of the salt
CNST_j	The jth fixed, i.e., constant, block.
K	A block cipher key.
$K1$	The first RMAC key.
$K2$	The second RMAC key.
$K3$	A key that is derived from the second RMAC key and the salt.
M	The message.
M_j	The j^{th} block in the partition of the padded message.
O_j	The j^{th} output block.
PAD	The padding that is appended to the message.
R	The salt.

4.2.2 Operations and Functions

0^s	The bit string consisting of s '0' bits.
$X \,\|\, Y$	The concatenation of two bit strings X and Y.
$X \oplus Y$	The bitwise exclusive-OR of two bit strings X and Y of the same length.
$CIPH_K(X)$	The forward cipher function of the block cipher algorithm under the key K applied to the data block X.
$MSB_s(X)$	The bit string consisting of the s most significant bits of the bit string X.
$RMAC(R,M)$	The RMAC message authentication code for message M with salt R.

5 Preliminaries

5.1 The Underlying Block Cipher Algorithm

The RMAC algorithm specified in this Recommendation depends on the choice of an underlying symmetric key block cipher algorithm; the RMAC algorithm is thus a mode of operation (mode, for short) of the symmetric key block cipher. The underlying block cipher algorithm must be approved, and two secret, random keys for the block cipher algorithm shall be established. The keys regulate the functioning of the block cipher algorithm and, thus, by extension, the functioning of the mode. The specifications of the block cipher algorithm and the mode are public, so the security of the mode depends, at a minimum, on the secrecy of the keys.

For any given key, the underlying block cipher algorithm of the mode consists of two processes that are inverses of each other. As part of the choice of the block cipher algorithm, one of the two processes of the block cipher algorithm is designated as the forward cipher function. The inverse of this process is called the inverse cipher function. Because the RMAC mode does not require the inverse cipher function, the forward cipher function in this Part of the Recommendation is simply called the block cipher.

5.2 Elements of RMAC

The block cipher keys that are required for the RMAC mode are bit strings, denoted $K1$ and $K2$, whose bit length, denoted k, depends on the choice of the block cipher algorithm. The keys shall be random or pseudorandom, distinct from keys that are used for other purposes, and secret. The two keys shall each be established by an approved key establishment method, or the keys shall be derived from a single key K, which is established by an approved key establishment method. A method for deriving $K1$ and $K2$ from a single, master key K is given in Appendix B.1.

The block cipher is a function on bit strings of a fixed bit length. The fixed bit length of the bit strings is called the block size and is denoted b; any bit string whose bit length is b is called a (data) block. Under a key K, the block cipher function is denoted $CIPH_K$.

For the AES algorithm, b=128, and k=128, 192, or 256; for Triple DES, b=64, and k=112 or 168.

The data to be authenticated is one input to the RMAC MAC generation function; the data in this context is called the message, denoted M.

Another input to the MAC generation function is a parameter associated with the message called the salt, denoted R. The role of the salt in the MAC generation function is to randomize (i.e., "flavor") the second key, K2. The bit length of the salt, denoted r, is determined by the choice of a parameter set that is specified in Section 6.2. The use of the salt is optional in the sense that a parameter set may be chosen in which r=0. When r?0, the method for generating the salt shall ensure that the expected probability of repeating the salt for different messages is negligible. The generation of the salt is discussed further in Appendix B.2.

The RMAC MAC generation function is denoted $RMAC$, so that the output of the function, the MAC, is denoted $RMAC(R,M)$. The bit length of the MAC, denoted m, is determined by the choice of a parameter set that is specified in Section 6.2. The authentication tag to the message is the ordered pair $(R, RMAC(R,M))$; thus, the tag consists of one part, the salt, that may be independent of the message and a second part, the MAC, that depends on both the salt and the message. The total number of bits in the tag is thus $r+m$.

5.3 Examples of Operations and Functions

For a nonnegative integer s, the bit string consisting of s ‘0’ bits is denoted 0^s.

The concatenation operation on bit strings is denoted ¦ ; for example, 001¦ 10111 = 00110111.

Given bit strings of equal length, the exclusive-OR operation, denoted ⊕, specifies the addition, modulo 2, of the bits in each bit position, i.e., without carries. Thus, 10011 ⊕ 10101= 00110, for example.

The function MSB_s returns the s most significant bits of the argument. Thus, for example, MSB_4(111011010) = 1110.

6 RMAC Specification

6.1 Message Formatting

The first steps of the MAC generation function are to append padding to the message and to partition the resulting string into complete blocks. The padding, denoted PAD, is a single ‘1’ bit followed by the minimum number of ‘0’ bits such that the total number of bits in the padded message is a multiple of the block size. The padded message is then partitioned into a sequence of n complete blocks, denoted $M_1, M_2, \ldots, M_n$. Thus,

$$M \,||\, PAD = M_1 \,||\, M_2 \,||\ldots||\, M_n \,.$$

If the bit length of M is a multiple of the block size, then $PAD = 1 \,||\, 0^{b-1}$, i.e., a complete block.

6.2 Parameter Sets

A parameter set is a pair of values for the bit lengths r and m of the two parts of the authentication tag, the salt and the MAC. The parameter sets for RMAC depend on the block size of the underlying block cipher algorithm. In Table 1, five parameter sets are given for the 128 bit block size, and two parameter sets are given for the 64 bit block size.

Table 1: Parameter Sets

Parameter Set	b=128		b=64	
	r	m	r	m
I	0	32	0	32
II	0	64	64	64
III	16	80	n/a	
IV	64	96	n/a	
V	128	128	n/a	

Although parameter set I offers the shortest authentication tags, it is not recommended for general use. The decision to use parameter set I requires a risk-benefit analysis of at least three factors: 1) the relevant attack models 2) the application environment, and 3) the value and longevity of the data to be protected. In particular, parameter setting I shall only be used if the controlling protocol or application environment sufficiently restricts the number of times that verification of an authentication tag can fail under any given pair of RMAC keys. For example, the short duration of a session, or, more generally, the low bandwidth of the communication channel may preclude many repeated trials.

Parameter sets II, III, IV, and V are appropriate for general use.

Some of the security considerations that underlie the selection of a parameter set are summarized in Appendix A. The expected work factors for important aspects of the attacks that are discussed in the appendix are summarized for each parameter set in Table 2 in Section A.4.

6.3 MAC Generation

The following is a specification of the RMAC MAC generation function:

```
Input:
block cipher $CIPH$;
block cipher keys $K1$ and $K2$ of bit length $k$;
parameter set $(r, m)$.
message M;
salt $R$ of bit length $r$.

Output:
message authentication code $RMAC(R, M)$ of bit length $m$.

Steps:
1.    Append to $M$ the padding string $PAD$, as described in Section 6.1.
2.    Partition $M \| PAD$ into $n$ blocks $M_1, M_2, \ldots, M_n$, as described in Section 6.1.
3.    $O_1 = CIPH_{K1}(M_1)$.
4.    For $j = 2$ to $n$, do $O_j = CIPH_{K1}(M_j \oplus O_{j-1})$ .
5.    If $r=0$, then $K3=K2$; else $K3 = K2 \oplus (R \| 0^{k-r})$.
6.    Return $RMAC(R, M) = MSB_m(CIPH_{K3}(O_n))$.
```

The calculations in Steps 3 and 4 are equivalent to encrypting the padded message using the cipher block chaining (CBC) mode of the block cipher [5], under the first RMAC key, with the zero block as the initialization vector. However, unlike CBC encryption, in which every output block from Steps 3 and 4 is part of the encryption output (i.e., the ciphertext), in RMAC, the output blocks in Steps 3 and 4 are intermediate results. In Step 6, the block cipher under a new key is applied to the final output block from Step 4, and the result is truncated as specified in the parameter set. The new key for this final application of the block cipher is obtained in Step 5 by exclusive-ORing the salt into the most significant bits of the second RMAC key.

The RMAC MAC generation function is illustrated in Figure 1.

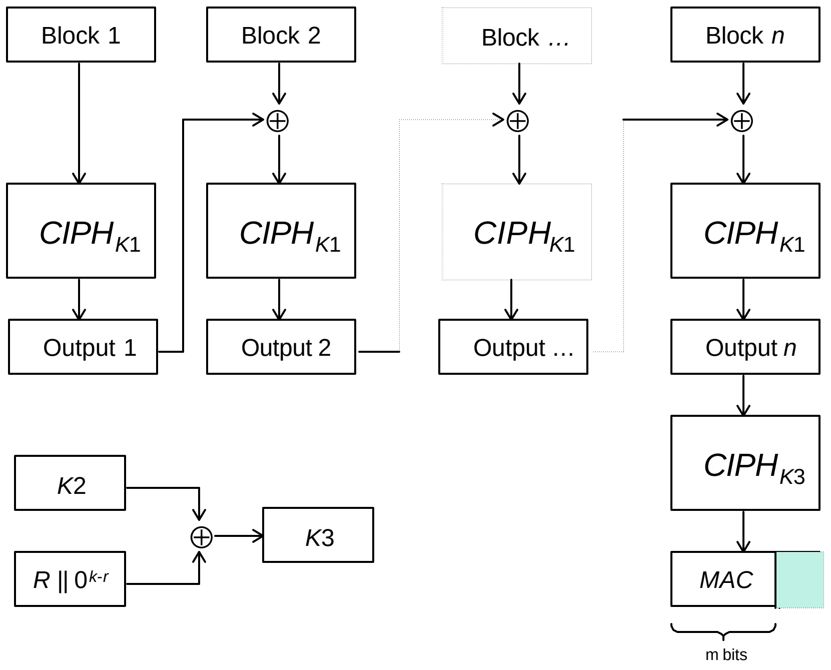

Figure 1: The RMAC MAC Generation Function

6.4 *Tag Generation and Verification*

The prerequisites for the authentication process are the establishment of an approved block cipher algorithm, two secret RMAC keys, and a parameter set[1] among the parties to the authentication of the data.

To generate an authentication tag on a message *M*, a party shall determine an associated salt *R* in accordance with Appendix B, generate *RMAC*(*R*,*M*), as specified in Section 6.3, and provide the authentication tag (*R*, *RMAC*(*R*,*M*)) to the data.

To verify an authentication tag (*R*', *MAC*'), a party shall apply the RMAC MAC generation function, as specified in Section 6.3, to the received message *M*' and the received salt *R*' within the tag. If the computed MAC, i.e., *RMAC*(*R*',*M*'), is identical to the received MAC, i.e. *MAC'*, then verification succeeds; otherwise, verification fails, and the message should not be considered authentic.

[1] For tag verification, the parameter set is implicit in the bit length of the tag.

Appendix A: Security Considerations

The submitters of RMAC present a security analysis of RMAC in [6]. In this appendix, three types of attacks on general MAC algorithms are summarized, and discussed with respect to RMAC: key search, general forgery, and extension forgery based on birthday collisions.

A.1 Exhaustive Key Search

In principle, given sufficiently many valid message-tag pairs, an unauthorized party can exhaustively search, off-line, every possible key to the MAC generation algorithm. After recovering the secret key, by this method or any other method, the unauthorized party could generate a forgery, i.e., a valid authentication tag, for any message.

The number of RMAC keys is so large that exhaustive key search of RMAC is impractical for the foreseeable future. In particular, for the key size k, which is at least 112 bits for the approved block cipher algorithms, the exhaustive search for the two RMAC keys would be expected to require the generation of 2^{2k-1} MACs. Even if the two RMAC keys are derived from a single master key, as discussed in Appendix B.1, the exhaustive search for the master key would be expected to require the generation of 2^{k-1} MACs.

A.2 General Forgery

The successful verification of a MAC does not guarantee that the associated message is authentic: there is a small chance that an unauthorized party can guess a valid MAC of an arbitrary (i.e., inauthentic) message. Moreover, if many message forgeries are presented for verification, the probability increases that, eventually, verification will succeed for one of them. This limitation is inherent in any MAC algorithm.

The protection that the RMAC algorithm provides against such forgeries is determined by the bit length of MAC, m, which in turn is determined by the choice of a parameter set. The probability of successful verification of an arbitrary MAC with any given salt on any given message is expected to be 2^{-m}; therefore, larger values of m offer greater protection against general forgery.

A.3 Extension Forgery Based on a Collision

The underlying idea of extension forgery attacks is for the unauthorized party to find a collision, i.e., two different messages with the same MAC (before any truncation). If the colliding messages are each concatenated with a common string, then, for many MAC algorithms, including RMAC, the two extended messages have a common MAC; therefore, the knowledge of the MAC of one extended message facilitates the forgery of the other extended message. The unauthorized party can choose the second part of the forged message, i.e., the common string, but generally cannot control the first part, i.e., either of the original, colliding messages.

In principle, collisions may exist, because there are many more possible messages than possible MACs. A collision may be detected by the collection and search of a sufficiently large set of message-MAC pairs. By the so-called "birthday surprise" (see, for example, [7]), the size of this sufficiently large set is expected to be, approximately, the square root of the number of possible MAC strings, before any truncation.

For RMAC, the extension forgery requires that the salt values, R, are the same for the two colliding messages, as well as the untruncated MACs, i.e., $CIPH_{K3}(O_n)$ in the specification of Section 6.3. Therefore, larger values of the block size, b, and the salt size, r, provide greater protection against extension forgery. In particular, the unauthorized party would have to collect at least $2^{(b+r)/2}$ message-tag pairs in order to expect to detect a collision.

Moreover, if a parameter set is chosen in which $m<b$, i.e., if $CIPH_{K3}(O_n)$ is truncated to produce the MAC, then the discarded bits may be difficult for an unauthorized party to determine, so collisions may be difficult to detect. Parameter sets in which $m<b$ may also provide some protection against other types of attacks.

A.4 Summary of Security Properties of Parameter Sets

In Table 2, the expected work factors for the important aspects of the attacks discussed in Sections A.1-A.3 are summarized for the RMAC parameter sets. The values for exhaustive key search are given for the case in which the two RMAC keys are generated from a single master key as discussed in Section B.1.

Table 2: Expected Work Factors for Three Types of Attacks on RMAC

RMAC Parameter Set	Exhaustive Key Search (MAC Generation Operations)	General Forgery (Success Probability for a Single Trial)	Extension Forgery (Message-Tag Pairs)
I	2^{k-1}	2^{-32}	2^{32} (b=64) or 2^{64} (b=128)
II	2^{k-1}	2^{-64}	2^{64}
III	2^{k-1}	2^{-80}	2^{72}
IV	2^{k-1}	2^{-96}	2^{96}
V	2^{k-1}	2^{-128}	2^{128}

Appendix B: The Generation of RMAC Parameters

B.1 Derivation of RMAC keys from a Master Key

The two secret RMAC keys, $K1$ and $K2$, may be derived from a single master key, K, in order to save bandwidth or storage, at the cost of extra invocations of the block cipher to set up the RMAC keys. For example, let CNST_1, CNST_2, CNST_3, CNST_4, CNST_5, and CNST_6 be constants, i.e., fixed, distinct blocks, and let k and b be the key length and block length of the approved block cipher, as before. If $k = 3b$, then $K1$ and $K2$ may be derived from the set of constants as follows:

$$K1=MSB_k(CIPH_K(\text{CNST}_1) \,||\, CIPH_K(\text{CNST}_3) \,||\, CIPH_K(\text{CNST}_5))$$
$$K2=MSB_k(CIPH_K(\text{CNST}_2) \,||\, CIPH_K(\text{CNST}_4) \,||\, CIPH_K(\text{CNST}_6)).$$

If $k=b$, then this definition reduces to $K1=CIPH_K(\text{CNST}_1)$ and $K2=CIPH_K(\text{CNST}_2)$, and thus only two .constants are actually required.

Similarly, if $b<k\leq 2b$, then the definition becomes $K1= MSB_k(CIPH_K(\text{CNST}_1) \,||\, CIPH_K(\text{CNST}_3))$ and $K2=MSB_k(CIPH_K(\text{CNST}_2) \,||\, CIPH_K(\text{CNST}_4))$, and thus only four constants are required.

B.2 Salt Generation

The salt values associated with messages shall repeat with no more than negligible probability. In particular, the expected probability that the same salt will be associated with two different messages that are authenticated under the scope of any pair of RMAC keys shall be no greater than for random values of salt. Therefore, one approach to meeting the requirement is to generate the salt by a deterministic approved random number generator.

Another approach is to ensure that the probability of associating the same salt to different messages is zero, in other words, to generate a nonce to be the salt. For example, the salt may be a counter or a message number.

Appendix C: Example Vectors for RMAC

[Vectors to be provided later.]

Appendix D: References

[1] É. Jaulmes, A. Joux, and F. Valette, “RMAC: A randomized MAC beyond the birthday paradox limit.” Available at http://csrc.nist.gov/encryption/modes/proposedmodes/.

[2] FIPS Publication 197, “Advanced Encryption Standard (AES).” U.S. DoC/NIST, November 26, 2001.

[3] FIPS Publication 46-3, “Data Encryption Standard (DES).” U.S. DoC/NIST, October 25, 1999.

[4] FIPS Publication 198, “The Keyed-Hash Message Authentication Code (HMAC).” U.S. DoC/NIST, March 6, 2002.

[5] Special Publication 800-38A, “Recommendation for Block Cipher Modes of Operation.” U.S. DoC/NIST, December 2001.

[6] É. Jaulmes, A. Joux, and F. Valette, “On the security of randomized cbc-mac beyond the birthday paradox limit: A new construction.” Available at http://eprint.iacr.org, January 15, 2002.

[7] A. Menezes, P. van Oorschot, and S. Vanstone, “Handbook of Applied Cryptography.” CRC Press, New York, 1997.